Security Working Group

Report on Trust Framework Technical Requirements for OAuth 2.0

Mohammad Jafari,
Kathleen Connor, John M. Davis, Christopher Shawn

Version 2.1

March 29, 2019

Table of Contents

List of Figures

List of Tables

1 INTRODUCTION

Establishing trust is often a prerequisite before two entities engage in any transaction; each party needs to develop some level of confidence about the identity and the reliability of the other party. The process which enables two parties to establish trust and enter into a trust relationship is referred to as the Trust Handshake.

As a broad umbrella term, Trust Framework refers to all the policy, process, and technical aspects of establishing and maintaining trust. A Trust Framework ensures that the parties have "continued confidence in one another" by "stipulating adherence to standards, formalizing assessment processes, and defining roles and responsibilities of multi-party arrangements [2]." Trust Framework covers "rights and responsibilities," "policies and standards," and "processes and procedures that provide assurance." [1]

More particularly, a Trust Framework includes the following components [1]:

- *System rules* defining the interactions between parties,
- A *legal structure*, which identifies the rights, responsibilities, and liabilities associated with different parties, and
- A structure for establishing and recognizing *conformance* across different parties.

From a different perspective, a Trust Framework must cover different aspects of establishing trustworthy *authentication*, *access control*, and *traceability* [3].

1.1 Scope of this Report

The focus of this report is identifying a set of technical requirements for establishing trust in an OAuth 2.0 ecosystem. Based on the different aspects of Trust Frameworks as mentioned above, Figure 1 puts the focus of this report in perspective.

This report will focus on technical requirements, so, it will not cover the legal structure, including policies, liabilities, responsibilities, and processes. From the conformance aspects, this report will only cover technical aspects of the conformance. Furthermore, since OAuth 2.0 is an authorization and delegation framework, the authorization aspects of the Trust Framework will be of primary focus.

	Authentication	**Authorization**	**Traceability**
System Rules			
Conformance			
Legal Structure			

Figure 1: High-level positioning of the focus of this report based on the broader aspects of Trust Frameworks

1.2 Structure of this Report

First, a brief overview of an OAuth ecosystem and existing trust relations is discussed in Section 2.

Section 3 presents a set of high-level requirements, followed by more specific technical requirements in Section 4. Table 1 summarizes how the technical requirements are related and aligned with the high-level requirements.

Section 5 provides an overview of some of the existing specifications and technologies which cover the broad requirements discussed in this report. Table 2 connects these existing technologies to the technical requirements and demonstrates what is and is not covered by the existing solutions, as the work on developing a more comprehensive trust solution in OAuth is still ongoing.

Finally, Section 6 provides a summary of the current gaps in the existing specifications and technologies and recommendations for future work.

2 TRUST RELATIONS IN AN OAUTH ECOSYSTEM

OAuth is a set of specifications which enable a Client to gain access to information residing on a Resource Server (RS) by interacting with an Authorization Server (AS) to obtain the required permissions in the form of an Access Token. Figure 2 shows the participants of an OAuth ecosystem and their trust relations alongside a brief overview of the protocol. There are three major trust relationships in an OAuth ecosystem as will be discussed in the rest of this section.

2.1 Trust Between the RS and the AS

The Resource Server's trust in the Authorization Server is the core of the OAuth framework which enables the RS to rely on the Authorization Server's decisions about a Client's access to the resources under control of the RS. Furthermore, in order to engage in communication with the AS and request for information, for example, in token introspection, the AS must be able to recognize the RS and establish a level of trust with it.

In the existing OAuth specifications, establishing trust between the AS and the RS is assumed to be manual and out of band. This is because:

- In the majority of existing systems, the AS and RS belong to the same organization and security domain, and their communications may take place using proprietary and internal technologies [4].
- Setting up an RS to rely on an AS, or setting up a new RS to rely on an existing AS is practically an infrequent event which does not need to be dynamic.

Thus, establishing trust between AS and RS has not been of much interest or attention in the OAuth framework, and therefore, this report will not focus on the requirements for this trust relation, although this case is very similar to the case of establishing trust between the Client and the AS.

Note that the close connection between AS and RS which makes the assumption of manual out-of-band trust setup between these two entities acceptable, is not an inherent part of the OAuth framework and is simply a result of the way the majority of existing systems have set up their OAuth environment. So, it is possible that this changes as new use-cases emerge which may lead to new paradigm of relationships between RS and AS. This is further discussed in Section 6.8.

2.2 Trust Between the Client and the RS

The interactions with the Client requires that the RS have trust in the Client. In OAuth, this trust relation is completely handled by delegation to the AS; the RS will trust any Client approved by the AS, and will rely on the Authorization Server's judgement. This is the core contribution of the OAuth framework which enables the RS to avoid the burden of establishing and maintaining a trust relationship with every individual Client and making authorization decisions about their requests.

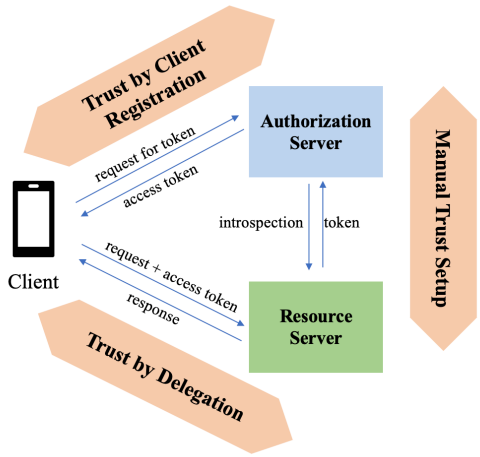

Figure 2: Overview of Trust Relations in an OAuth Ecosystem

2.3 Trust Between the Client and the AS

The AS must establish some level of trust in the Client before the Client is allowed to interact with the AS, for example, to ask for an access token. In small environments where the number of Clients is small, this can take place manually and statically by an admin adding a Client to a database of trusted Clients. But where there is a large number of Clients, manual setup will not be practical or efficient, and therefore a mechanism is required for establishing trust dynamically with the Client. This is called the Dynamic Client Registration [5].

The registration is essentially a trust handshake in which the Client engages in a protocol to establish trust with the server. This is the prerequisite to any future interactions with the OAuth server, for example, requesting an access token.

The Client registration request often includes a range of supporting credentials to make the case and convince the AS to trust the Client, for example, an attestation certifying Client's compliance with certain requirements. If successful, the AS notifies the Client about the success of the handshake, alongside other information such as the terms, conditions, and attributes of the trust and identifiers or credentials issued by the AS.

2.4 Client Registration Use-Case

Based on the above discussion, the main use-case for trust handshake, which will be the primary focus of this report, is narrowed down as the following: A Client, either completely unknown to the AS, or known to a limited extent, needs to establish trust with an OAuth Authorization Server to enable future communications with that AS.

The Client usually provides different types of credentials to support its case in requesting trust from the AS. The Client may engage in prior interactions with the AS or other parties to obtain such credentials before the time of registration.

As a result of the successful registration, the AS may need to issue credentials for the Client, such as a shared secret, to be able to facilitate further interactions with the Client.

3 HIGH-LEVEL REQUIREMENTS

The high-level requirements for Trust Framework solutions in OAuth are discussed in this section. These broad and general requirements guide the more specific technical requirements discussed in the following section and put them in perspective.

3.1 Scalability

The trust handshake process between the Clients and the AS must be capable of keeping up with the growing number of Clients and the growing frequency of registration.

With the emergence and prevalence of mobile applications, Trust Frameworks must support an increasing number of Clients with a corresponding high frequency of trust handshakes. Manual, and inherently slow, trust handshake processes can only support a limited number of Clients and a relatively low frequency of new registrations and cannot scale to meet this demand. While rigorous manual processes are still relevant, they cannot be the only way to establish trust as they cannot scale to accommodate the emerging application.

Scalability is the core requirement for an OAuth trust framework and most of the other high-level requirements mentioned in the subsequent subsections are either related to or further support scalability.

3.2 Separation of Concerns

Trusting a Client often depends upon compliance of the Client with various standards and requirements. As these compliance requirements increasingly grow more complicated and specific, verifying compliance of a Client requires specific expertise and specialization which is often outside the core business of the organization to which the AS belongs. Therefore, as a broad requirement, organizations must be able to outsource and offload such processes to external parties which specialize in such services and rely on a marketplace of third-party *Endorsers* which issue assertions about Clients.

This is also beneficial for the Clients, since instead of engaging in one-to-one processes for certification with every organization, with potentially different workflows and paperwork requirement, Clients can engage once with a standard Endorser and then re-use the certified credentials in establishing trust with different organizations.

3.3 Interoperability

Establishing trust should follow a similar and predictable process for different Authorization Servers across different organization; the Client must not be forced to pursue a different and custom process with every AS.

Interoperability is a broad requirement and for an OAuth ecosystem and the primary goal of the OAuth specifications is to define such uniform and standard mechanisms for the interactions between Clients and Authorization Servers. There are various aspects of interoperability which are not currently covered by the specifications as this report will later discuss, such as the interoperability requirements in relationships with Endorsers, or between the exchange of audit records between the RS and the AS.

3.4 Trust Spectrum

Authorization Servers must be able to recognize and define types, levels, or modalities for the trust in a Client; in other words, the trust decision must go beyond the yes-or-no binary and include more details about the conditions and attributes of trust.

As the application space grows and different types of Clients emerge, organizations need to support different types of trust for different types of Clients. So, the trust relation between a Client and an AS could be subject to attributes, conditions, rather than a binary state in which either there is either complete trust or no trust at all. A diverse set of trust relationships is important in addressing a diverse set of Clients, applications, and use-cases.

Trust attributes also enable a Client to bootstrap a trust relationship by starting from a basic level of trust and engaging in further trust handshakes to change the level or type of the trust, in a process known as Trust Elevation.

3.5 Rule-Based Trust Decisions

Authorization Servers must be able to make their trust decisions based on well-defined rules. The AS must be able to receive a request and rely on such rules to determine whether or not to trust the Client and to what extent.

Rule-based decisions enable a predictable process for approving a Client and are a requirement for automating the trust decisions.

4 TECHNICAL REQUIREMENTS

Table 1 provides a summary of the technical requirements and their alignment with the high-level requirements discussed in the previous section. The details of these requirements are discussed in the rest of this section.

These requirements are meant to be a general set, so, some systems may not need to support some of these requirements depending on other high-level requirements and the specific use-cases.

Table 1: Summary of technical requirements and their connection to high-level requirements

		High-Level Requirements				
		Scalability	Separation of Concerns	Interoperability	Trust Spectrum	Rule-Based Decisions
Discovery	The AS should enable Clients to discover the endpoint and the requirements for a trust handshake.			■		
Client Types	The AS should be able to recognize different types of Clients or Clients representing more than one type of entity (e.g., a human user and a device).	■		■		■
Handshake Types	The AS should support static trust handshake by manual addition of a Client to the database of trust.					■
	The AS should support dynamic trust handshakes which include manual steps to be approved by a human agent.					■
	The AS should support dynamic trust handshakes which enable the Client to establish trust by interacting with a dynamic registration endpoint.	■		■		■
	The AS should support ad-hoc trust handshakes alongside an authorization token request.	■		■		■
Credentials	The AS should be able to accept and process self-certified claims by a Client.	■		■		■
	The AS should be able to accept and process claims issued and certified by trusted third-party Endorsers.	■	■	■		■
	The AS should be able to process manual credentials and issue electronic assertions based on their content.	■		■		■
	The AS should be able to accept and process manual credentials in a manual trust handshake process.					■
Handshake Results	The AS should be able to include advice when rejecting a trust handshake request.	■		■		■
	The AS should be able to accept a Trust Handshake request conditionally.	■				■
Trust Attributes	The AS should be able to record and consider various types or levels of trust based on trust attributes.	■			■	■

	The AS should be able to assign scopes to a trust relationship as broad restrictions applicable to all future interactions.					
	The AS should be able to issue Client credentials capable of recording level or attributes of trust.					
	The AS should enable the Client to re-engage with the AS to change the level or attributes of the trust relation.					
Authentication	The AS should be able to issue authentication credentials for the Client as a result of successful trust handshake.					
Audit	The AS should record, trace, and enable access to records of Client's trust handshakes and subsequent interactions.					

4.1 Discovery

The AS should enable Clients to discover the endpoint and the requirements for a trust handshake.

Discovery mechanisms improve interoperability by giving Clients a uniform way of interacting with the AS to find the requirements and endpoints for trust handshake. Without discovery, the Client has to examine manual documents for interacting with different Authorization Servers.

The discovery should include the registration endpoint used for trust handshake, and other metadata about the server such as the required credentials.

4.2 Client Types

The AS should be able to recognize different types of clients or clients representing more than one type of entity.

Depending on the applications or use-case, a Client may correspond to different types of entities, for example, an organization, a machine or device, a human user, a software, or a software instance, as will be discussed later below. Note that these categories are not mutually exclusive, and a Client may represent an intersection of a number of these entities. For example, a Client may represent an instance of software developed by an organization and installed on a specific device.

Recognizing and supporting different types of entities enables the AS to follow different rules based on the Client types in making rule-based decisions about trust.

Differentiating between different Clients also enables the AS to operate in a more scalable way by following more flexible trust requirements for Client types that correspond to a large number of Clients. For example, when a Client represents an entire organization, establishing trust can enable a broad set of transactions in the future but the frequency of registering new Clients is not high; therefore, the requirements for establishing trust with this client could be more rigorous and costly. In contrast, when the Client represents a software application installed on a mobile device, the number of such Clients and the frequency of new registrations is much higher but the scope of future transactions is much more limited; therefore, a more flexible set of requirements can be applied.

4.2.1 Organization

A Client could represent an entire organization. In such cases, if the trust handshake is successful, it will cover any future interaction between the two organizations. In this case, Client credentials are associated with the organization.

Establishing trust between two organizations is often an elaborate process which takes manual engagement and agreements. This is reasonable since establishing trust across organizations usually covers a potentially broad range of future transactions. Moreover, such cross-organizational trust handshakes are usually not very frequent, so, a process involving manual steps does not pose a significant scalability risk.

4.2.2 Machine or Device

The Client may represent a particular, virtual or physical, machine or device. In this case, credentials are associated with the specific machine (e.g., its network address, domain name, device unique identifier, or other identifiers).

4.2.3 Human User

The Client may represent a human user. In this case, credentials are associated with the identity of the person (e.g., an email address, driver's license, etc.).

4.2.4 Software

Especially in the case of mobile or web applications, a Client may represent a particular software application. In this case, credentials correspond to the software and its capabilities, for example, certification of compliance to certain standards. This enables the AS to distinguish between different software applications and establish trust with only those that meet certain criteria.

It is also possible for an AS or an Endorser to issue credentials, such as unique identifiers or shared secrets, for a particular software, and make it available to the vendor as part of a manual out-of-band process so that the software or its instances can be identified at the time of trust handshake based on these credentials.

4.2.5 Software Instance

In some use-cases, a software application as a general entity is distinguished from its specific instances installed or deployed on specific servers or devices. When a software application is installed on a specific device, or deployed in a specific environment (e.g., a particular data center), it constitutes a separate, more specific entity which is different from the software application itself. The attributes of a specific instance of software installation are sometimes important and consequential in establishing trust and the AS may want to trust one instance of a software application while not trusting another instance. For example, a web application may be trusted when deployed on a server with Transport Layer Security (TLS) enabled while the same application may not be trusted when deployed on a different server with different configuration. Or, a mobile application may be trusted if installed on an approved device, while another instance of the same software, installed on a different device may not meet the trust requirements.

4.3 Trust Handshake Types

This group of requirements specify different types of trust handshakes which an AS must support.

4.3.1 Static Trust Handshake

The AS should support static trust handshakes by manual addition of a Client to the database of trust. Static handshake is a manual process in which a Client's credentials are inspected and if approved, the Client identifier is manually added to a database of trusted entities.

In cases where establishing trust requires manual steps for evaluating credentials, or legal negotiations, etc. the handshake is inherently static. The static nature of the protocol implies that the list of trusted entities is managed manually, and it is not possible for a new Client to establish trust automatically.

4.3.2 Dynamic Trust Handshake with Manual Steps

The AS should support dynamic trust handshakes which include manual steps to be approved by a human agent.

The dynamic trust handshake endpoint provides a mechanism for Clients to contact the AS and submit credentials and other information supporting a request to establish trust. The AS should be

able to support manual steps (e.g., manual verification of certifications), in the process of approving the trust handshake. For example, a human agent must be able to review a submitted certification and make a judgment on whether that is accepted.

At the technical level, the protocol will be asynchronous and either the Client has to come back and check the status of the request, or provide a callback Application Programming Interface (API), via which the AS sends the Client a notification about the result of the trust handshake.

4.3.3 Dynamic Trust Handshake

The AS should support dynamic trust handshakes which enable the Client to establish trust by interacting with a dynamic registration endpoint.

Using a dynamic trust handshake endpoint, a Client without prior relationship with the AS, can submit a request with all the supporting credentials. The AS, then, makes an automated rule-based decision, and responds to the Client with the result of the handshake.

4.3.4 Ad-hoc Trust Handshake

The AS should support ad-hoc trust handshakes alongside an authorization token request.

Ad-hoc trust handshake enables a Client to hand in credentials at the time of requesting for authorization. In other words, evaluating the Client's trustworthiness happens at the same time as the authorization for a requested transaction, such as requesting an access token.

This approach is very flexible and enables any Client to establish trust for the span of one transaction right around the time of that transaction. However, it is only efficient for one-off transactions or when establishing trust handshake is not a costly process. If the Client intends to engage in many future transactions, and especially when the establishing trust involves costly steps such as manual reviews, ad-hoc trust handshakes are not practical.

4.4 Credentials

The credentials presented by a Client in support of a request for establishing trust can take different forms. The type of acceptable credentials affects the way they can be verified and therefore it is consequential in the entire trust handshake process. This section discusses the requirements for supporting different types of credentials.

Interoperability is a particularly crucial requirement in issuing and processing credentials. Clients must be able to use credentials certified by self or by other Endorsers with different Authorization Servers without having to acquire or modify them per-AS. Without standard and mutually understandable formats and vocabularies, it will not be possible for an AS to consume assertions issued by third-party Endorsers, or the Client will have to generate different assertions by following a different convention and vocabulary for every different AS.

4.4.1 Self-Certified Claims

The AS should be able to accept and process self-certified claims by a Client.

Self-certified claims are assertions by the Client which are not further certified by any other party. Such claims are either signed by the Client using the Client's public key or a shared secret, or simply asserted as part of the request in the secure protocol over which the trust handshake takes place (i.e., Hypertext Transfer Protocol Secure (HTTPS)).

Accepting such claims are based on some basic level of *a priori* trust by the AS in the Client and the Client's reliability in making such claims, so, such claims are often used alongside other credentials which are certified by other parties.

Moreover, some self-certified claims are only recorded for the purpose of accountability and auditing. For example, if a Client representing a mobile application makes a self-certified assertion about accepting the terms of service of the AS, or compliance with its privacy policy, this assertion can be recorded and used as evidence to take punitive actions in case of a violation.

4.4.2 Claims Certified by Trusted Endorser

The AS should be able to accept and process claims issued and certified by trusted third-party Endorsers.

Trusted third-parties, also known as Endorsers or Endorsing Bodies (EB) are entities specialized in evaluating credentials. Such entities can establish trust with the AS using out of band mechanisms, so that the AS can rely on their assertions when presented by a Client.

For example, a mobile app auditor may have an elaborate code inspection process to certify compliance with some standards. The results of this process are expressed in the form of a signed assertion. The Client can, then, include this assertion, or a link to it, when engaging in a trust handshake with different Authorization Servers which trust that Endorser.

Relying on Endorsers is an important step in enabling a scalable approach to trust handshake by enabling third-parties to handle the workload of the certification process. It also improves the separation of concerns by allowing the organization in charge of the AS to focus on its core business and outsource the certification process to specialized third-parties.

Defining a standard registry and vocabulary for assertions expressed by Endorsers is a key step in guaranteeing interoperability so that assertions issued by Endorsers are universally understood across different Authorization Servers, and ensuring that Clients can re-use such credentials with different Authorization Servers without having to go through custom processes for each AS.

4.4.3 Manual Credentials

The AS should be able to accept and process manual credentials in a manual trust handshake process.

The AS should be able to process manual credentials such as a scanned copy of a certification of compliance. This will force the handshake process to be static or a dynamic handshake with manual steps to evaluate and approve the manual credentials.

4.4.4 Turning Manual Credentials to Electronic Assertions

The AS should be able to process manual credentials and issue electronic assertions based on their content.

Turning manual credentials into electronic assertions can enable the Client to submit manual credentials out of band, receive a signed assertion as a result and proceed to a dynamic trust handshake with the AS later. This enables Clients to be able to re-use the electronic assertions in future trust handshakes with the same AS without falling back to waiting for a manual process. In environments where it is common to repeat the trust handshake (e.g., to change trust attributes or level of trust), this improves the scalability of the trust handshake process by eliminating future manual steps.

Using a standard registry and vocabulary improves the interoperability by enabling the Client to potentially use such assertions with other Authorization Servers as a form of credentials certified by a third-party.

4.5 Handshake Results

This group of requirements address the types of result that the AS must support when handling the handshake request by the Client.

4.5.1 Rejection with Advice

The AS should be able to include advice when rejecting a trust handshake request.

When the Client's request to establish trust is declined, the AS must be able to provide the information about why the request was rejected and what the Client should do to meet the requirements.

Having standard forms for communicating this information to the Client is important from a scalability perspective since it enables Clients to automatically process these cases without manual inspection or out-of-band communication. It is also important from interoperability perspective that the Clients are able to process these cases in a uniform way that works across different Authorization Servers.

4.5.2 Conditional Acceptance

The AS should be able to accept a Trust Handshake request conditionally.

Conditional acceptance enables the AS to inform the Client that it has accepted the trust handshake on the condition that additional requirements be met by the Client. A common example is confirming the ownership of an email address or submission of additional credentials.

Conditional acceptance is more complex to implement since it has to handle the state where a Client's trust request has been accepted but has not met the final conditions yet. This complexity is only justified when evaluation of the initial trust request is too expensive to be repeated. For example, if a Client has already passed the manual steps of approval, it makes sense to avoid repeating those steps and issue a conditional acceptance.

4.6 Trust Attributes

This group of requirements address the attributes and properties of a trust relation between the AS and the Client and the Authorization Server's ability to determine and record them.

Trust attributes are the properties associated with a trust relationship which specify the characteristics and modalities of the trust. Instead of a binary, all-or-nothing approach to trust, the AS is able to rely on various distinct types, or levels of trust by determining the parameters applicable to the trust relationship. The most common example of trust attributes is the validity period which narrows down the period in time in which the trust relation is valid.

Some trust attributes, such as scopes or validity period, are consequential in the authorization of subsequent interactions with the Client as they lay out the context and boundaries in which the Client is considered trusted. Some trust attributes, however, may only be used for the purpose of audit record and accounting.

Trust levels encapsulate a group of trust attributes in the form of a number, a category, or a vector of values. This abstraction makes it easier to reference the trust attributes in other policy rules such as authorization decisions. Moreover, from a software-engineering perspective, it helps

the AS to transparently change the details of the attributes associated with the level of trust without imposing any external change to the Clients or other applications.

This is similar to the idea of *levels of assurance* which has been around in the identity proofing and authentication literature in the form of Level of Assurance, as proposed by the earlier version of NIST Special Publication 800-63 [6] or the more recent specifications for Vectors of Trust [7].

4.6.1 Levels or Types of Trust

The AS should be able to record and consider various types or levels of trust based on trust attributes.

A rule-based approach to trust decisions must enable the AS to determine the attributes of the trust at the conclusion of a trust handshake. The AS must be able to record, maintain, and when necessary, update or delete these records, using a database or a server-side data structure.

Determining and recording different types or levels of trust based on trust attributes also provides more flexibility for the AS to establish different types of trust with different parameters based on the requirement of various use-cases. This ultimately improves the scalability of the trust handshake process.

4.6.2 Trust Scopes

The AS should be able to assign scopes to a trust relationship as broad restrictions applicable to all future interactions.

A scope is a form of trust attribute which narrows down the area in which a trust relationship has been established. For example, based on the credentials provider by a Client, the AS may determine that its trust in the Client is restricted to a specific server endpoint only, such as the token endpoint. Trust scopes essentially specify broad restrictions applicable to the trust relationship which should be taken into account in all future interactions with the Client.

In the OAuth framework, the broader trust scopes assigned to a Client at the time of trust handshake can guide and restrict the authorization scopes which can be granted to the Client in its future token requests.

4.6.3 Recording Trust Attributes in Client Credentials

The AS should be able to issue Client credentials capable of recording level or attributes of trust.

In cases where the AS issues Client credentials (as discussed later in Section 4.7), the capability of the AS to securely record the trust attributes within the Client's credentials facilitates recognition of the Client and the attributes corresponding to the trust relationship between the Client and the AS in future interactions.

This improves the performance in environments where such Client interactions are frequent and ultimately improves scalability. Trust attributes are usually recorded and maintained by the AS using a database or a server-side data structure. This gives more control to the server in use-cases such as revoking trust, but since it relies on a central trust database which must be queried in every transaction with Clients, it can become inefficient and a performance bottleneck as the number of Clients grow. Recording trust metadata in the Client credentials, on the other hand, helps the AS to recognize a Client and its scope and level of trust more efficiently in future interactions.

Recording attributes in client credentials requires using complex credential structure capable of storing metadata in the form of a JavaScript Object Notation (JSON) Web Token (JWT).

4.6.4 Trust Elevation

The AS should enable the Client to re-engage with the AS to change the level or attributes of the trust relation.

In cases where an existing Client needs to chance the level or type of trust with the AS, trust elevation enables building upon an existing trust rather than starting over. This is particularly important in cases where the initial trust handshake is costly, for example, when manual steps are involved. Ultimately, this furthers the idea of trust spectrum and improves the scalability of the trust handshake process by eliminating repetitive and redundant steps.

4.7 Authentication

The AS should be able to issue authentication credentials for the Client as a result of successful trust handshake.

These credentials are used by the Client in future interactions with the AS to identify the Client. The existence and validity of these credentials is indicative of an existing trust relationship with the Client. This ultimately improves the scalability since the AS will have a straightforward mechanism for authenticating the Client and determining its level or type of trust in future interactions, using its own issued credentials and without having to process and validate credentials issued by other parties.

Client credentials can be as simple as a shared secret key between the AS and the Client. Often, the Client credentials issued by the server are more complex and include a set of metadata signed by the server, for example, in the form of a JSON Web Token (JWT) as discussed earlier.

4.8 Audit

The AS should record, trace, and enable access to records of Client's trust handshakes and subsequent interactions.

Recording audit logs is a common requirement for different services. The AS serves an ecosystem which may include several Resource Servers which rely on the AS for authorization decisions. So, the records of trust handshake and the ability to trace back authorization decisions about a Client to the record of that Client's trust handshake is of interest both for the AS and the relying Resource Severs. Therefore, a standard and interoperable mechanism for creating and enabling access to such records is required.

5 EXISTING SOLUTIONS

There are a number of existing protocols and specifications which cover parts of the broad set of requirements laid out in this report. A comprehensive set of specifications for trust framework in OAuth, however, is still a work in progress and there are ongoing efforts, at the time of writing, to specify the requirements and to develop new specifications to address them, most notably, the draft document for Scaling Dynamic Client Registration and Endpoint Discovery for Open APIs under consideration by the Health Level Seven International (HL7) Fast Healthcare Interoperability Resources (FHIR) community [8].

This section provides a brief overview of the existing solutions and discusses the trust framework requirements they cover. A summary of these solutions vis-à-vis the requirements is depicted in Table 2.

Table 2: Summary of the requirement coverage by the existing solutions (light shade: brief or indirect support; dark shade: focused on addressing the requirement)

		Existing Solutions/Specifications						
	Core OAuth	Server Metadata	Dynamic Client Registration	Registration Management	POET	XSPA SAML Profile 2.0	Community Proposal	FHIR
The AS should enable Clients to discover the endpoint and the requirements for a trust handshake.								
The AS should be able to recognize different types of Clients or Clients representing more than one type of entity (e.g., a human user and a device).								
The AS should support static trust handshake by manual addition of a client to the database of trust.								
The AS should support dynamic trust handshakes which include manual steps to be approved by a human agent.								
The AS should support dynamic trust handshakes which enable the Client to establish trust by interacting with a dynamic registration endpoint.								
The AS should support ad-hoc trust handshakes alongside an authorization token request.								
The AS should be able to accept and process self-certified claims by a Client.								
The AS should be able to accept and process claims issued and certified by trusted third-party Endorsers.								
The AS should be able to process manual credentials and issue electronic assertions based on their content.								
The AS should be able to accept and process manual credentials in a manual trust handshake process.								
The AS should be able to include advice when rejecting a trust handshake request.								
The AS should be able to accept a trust handshake request conditionally.								
The AS should be able to record and consider various types or levels of trust based on trust attributes.								
The AS should be able to assign scopes to a trust relationship as broad restrictions applicable to all future interactions.								
The AS should be able to issue Client credentials capable of recording level or attributed of trust.								
The AS should enable the Client to re-engage with the AS to change the level or attributes of the trust relation.								
The AS should be able to issue authentication credentials for the Client as a result of successful trust handshake.								
The AS should record, trace, and enable access to records of Client's trust handshakes and subsequent interactions.								

5.1 OAuth 2.0 Authorization Server Metadata

The OAuth 2.0 Authorization Server Metadata defines an extensible set of standard metadata which is published by an AS at a well-known (i.e., fixed) endpoint and provides general information about the AS [9]. This metadata which can be signed by the AS in the form of a JSON Web Token (JWT), provides a range of information about the AS, including its Client registration endpoint, the endpoint for engaging in a trust handshake.

The standard registry for this metadata does not include any fields for specifying the trust handshake requirements by the AS, however, since it is an extensible registry, new values can be defined by other specifications to cover other trust-related metadata such as required credentials.

5.2 OAuth Dynamic Client Registration

The Dynamic Client Registration specifications for OAuth 2.0 [5] define a protocol for a Client to establish trust with an AS by engaging in a dynamic trust handshake with the AS. The Client can be completely unknown to the server, or it may already have engaged in an out-of-band basic form of relationship beforehand, such as signing up on a web portal, leading to issuing an initial set of credentials for the dynamic client registration endpoint.

The protocol simply starts by the Client providing a set of self-asserted metadata as well as an optional set of assertions in the form of a JSON Web Token (JWT) signed by a third-party as will be discussed further below.

If successful, the server responds by issuing the Client an identifier which enables the Client to identify itself later when interacting with the authorization endpoint of the AS. This client identifier should be random and un-guessable if it is the only means for recognizing the Client in later interactions. Optionally, the server can issue the Client a secret which the Client must subsequently use to identify itself to the server, alongside the issued identifier. The secret may have an expiration date.

These specifications also define a set of standard metadata which can be provided by the Client as assertions to support its request for trust. The most important self-certified assertions by the Client are:

- A link to the *terms of service* document agreed to by the end-user represented by the Client.
- A link to the Client's *privacy policy*.
- A link to, or inline copy of, the Client's public key in the form of a JSON Web Key (JWK) object.
- The unique un-guessable identifier and version of the software application the Client represents. This identifier is the same across multiple instances or deployments of the same software product. It is issued out-of-band by the AS to software developers or vendors and enables the AS to recognize which software application is "running" behind this Client.

The Client can also provide a set of assertions about the software it represents signed by a trusted third-party in the form of a *Software Statement*. The Software Statement includes the software identifier as well as the identifier assigned by the software developer to the Client.

5.3 Dynamic Client Registration Management

The OAuth 2.0 Dynamic Client Registration Management Protocol [10] complements the dynamic Client registration by enabling the Client to access, update, or delete the registration

record of that Client, which is the metadata recorded by the AS about the Client at the time of registration.

This protocol is focused on application-related attributes for Client registration (e.g., redirect URLs), nevertheless, it defines a mechanism for the Client to access such attributes and update or delete them. This can be used by a Client to take note of the level of trust or trust attributes granted after a trust handshake and facilitate engaging in trust elevation in case the Client finds the level of trust or trust attributes inadequate.

5.4 Pre-OAuth Entity Trust (POET)

Pre-OAuth Entity Trust is a set of draft specifications proposed by the Blue Button 2.0 initiative of the Centers for Medicare and Medicaid Services (CMS) [11]. POET builds upon the OAuth Dynamic Client Registration specifications by defining a standard format for assertions issued by an Endorsing Body (EB) about Clients and the software they represent in the form of JSON Web Tokens (JWT). These assertions are very similar to those in the Software Statement in the OAuth Dynamic Client Registration protocol [5] and are specific to the software application represented by the Client.

These specifications also define extensions to the OAuth Dynamic Client Registration specifications for a Client to submit these assertions in its registration request. These specifications additionally require the AS to show the Client's submitted endorsements to the end-user in the interface for granting authorization the Client, later in the OAuth process. This enables the end-users to also consider such endorsements in their decisions whether or not to allow the application represented by the Client to access their data.

5.5 XSPA Profile for SAML v 2.0

The latest draft of OASIS Cross-Enterprise Security and Privacy Authorization (XSPA) profile for Security Assertion Markup Language (SAML) [12] includes a non-normative reference to the trust handshake use-case in healthcare. While the profile primarily defines the attributes commonly needed in authorizing an exchange transaction in the healthcare sector, it also defines broader attributes used in the trust handshake, including *certification* and *policy attestation*. This version of XSPA profile for SAML also defines a non-normative JSON encoding for these attributes which enables using them directly in a JSON Web Token (JWT) in any of the, above, or other similar protocols.

5.6 FHIR Community Proposal for Scaling Dynamic Client Registration and Endpoint Discovery

This draft proposal which is written and distributed by some of the application developers in the FHIR community, aims at covering some of the gaps and shortcomings in the existing OAuth specifications with a focus of scalability and interoperability. More specifically, it seeks to enable Authorization Servers to rely on a marketplace of known trusted Endorsers who can issue certifications about applications, as well as improving the process for the discovery of application, Endorsers, and AS endpoints.

This proposal does not currently include any specific solutions, but it identifies some of the important existing gaps.

6 EXISTING GAPS AND FUTURE WORK

This section highlights some of the important gaps in the requirements which are not currently covered by the existing specifications and technical solutions. Some of these gaps are already identified by the community and there is work in progress to cover them as discussed in Section 5.6.

6.1 Third-Party Assertions and Trusted Endorsers

The current specifications define some limited capabilities for the AS to rely on assertions issued by third-parties and lay out the base line for how assertions can be issued, signed, and recognized by an AS during a Trust Handshake. However, this is still far from a comprehensive framework for a marketplace of trusted Endorsers which Authorization Servers can rely on. A comprehensive solution would need to cover the following:

- A mechanism for universal identification of Endorsers.
- A mechanism for an AS to declare which Endorsers they trust so that Clients can ensure they obtain the right credentials and assertions.
- A standard registry and vocabulary for compliance and certifications so that Endorsers can issue assertions in a way that is meaningful and consumable to Authorization Servers.

6.2 Trust Requirements Metadata

The existing specifications do not define any standard metadata for an AS to announce its expectations for a successful trust handshake. This can be added to the existing metadata registry defined by the OAuth 2.0 Authorization Server Metadata [9]. Some examples for such metadata are:

- Levels or types of trust supported by the AS,
- Required credentials and certifications for each type or level of trust, and
- Trusted Endorsers whose assertions will be accepted by the AS.

6.3 Conditional Registration

The existing specifications do not offer a mechanism for conditional registration so that a Client can further interact with the AS to complete its registration process. This is important when the initial registration has already passed some costly steps such as manual reviews; repeating those steps is time-consuming and inefficient.

The Dynamic Client Registration Management specifications [10] have already defined a mechanism for Clients to engage in further interactions with the AS to inquire or update the state of the registration. Therefore, this requirement can be accommodated by extending these specifications to enable the Client to query the AS about any remaining conditions which must be met before activating the registration and using the update endpoint to add or update further credentials to the record if needed.

6.4 Advice on Failed Registrations

The existing specifications do not cover a standard way of returning informative advice to the Client in case a Trust Handshake fails based on policy rules. It is clearly possible for an implementation to use custom error messages to communicate such information, but these will be

custom and proprietary, and Clients will not be able to rely on them across different Authorization Servers.

6.5 Trust Scopes

Although scopes are an essential part of the OAuth authorization framework, current specifications do not define any mechanisms for assigning scopes to a trust relation, and relying on such scopes in other OAuth protocols.

Assigning scopes to the Client at the time of registration provides a mechanism to ensure that authorization scopes granted to a Client are bound by the trust scope associated with that Client. The AS can use this to exercise broad control and apply restrictions, based on policies, on what the Client can be authorized to do in future and what scopes it can be granted in future requests for access tokens.

For example, if the AS determines at the time of registration that the Client is only trusted for read transactions, this can be recorded as a `read/*` trust scope to ensure that all scopes subsequently granted to this Client in future access token requests must fall within this broader pattern and, for example, the AS never grants a `write/Medication` scope to this Client.

6.6 Trust Elevation

Current specifications do not define a mechanism for an existing registered Client to engage in a Trust Elevation protocol and change the level of trust or trust attributes. At the moment, in such cases the Client will have to start over by engaging in a brand-new trust handshake in order to change the level of trust or trust attributes, but there is no way for the Client to build upon an existing trust relation, for example, by presenting additional certifications or credentials. The Dynamic Client Registration Management specifications [10] lays out the ground work for subsequent interactions of a Client with the AS *after* establishing trust; so, a great starting point will be extending these specifications to accommodate the trust elevation use cases.

6.7 Audit

Audit and the ability to access audit records in a standard and inter-operable way is one of the pillars of Trust Frameworks as mentioned in Section 1. The OAuth specifications at the moment do not cover any standards or mechanisms or for recording and communicating audit events within the OAuth ecosystem. Defining a standard format and standard, discoverable endpoints for retrieving audit logs of Client registration is a future work.

6.8 AS-RS Trust Handshake

As discussed in Section 2, the OAuth framework is mostly silent about establishing trust between the Resource Server and the Authorization Server and this trust relation is assumed to be set up manually and out of band. This is mostly based on the assumption that the two servers are controlled by the same organization and there is not much dynamism in their relationship.

While these assumptions are true in most existing applications, they could be challenged by the emerging use-cases for relying on third-party OAuth Authorization Servers in cases like patient consent where an AS is a separate entity in charge of enforcing some policies. Enabling more scalable and interoperable ways for establishing trust between AS and RS is, therefore, an area of future work for OAuth Trust Frameworks.

7 ACRONYMS

API	Application Programming Interface
AS	Authorization Server
CMS	Centers for Medicare and Medicaid Services
EB	Endorsing Body
FHIR	Fast Healthcare Interoperability Resources
HL7	Health Level Seven International
HTTPS	Hypertext Transfer Protocol Secure
JSON	JavaScript Object Notation
JWK	JSON Web Key
JWT	JSON Web Token
POET	Pre-OAuth Entity Trust
RS	Resource Server
SAML	Security Assertion Markup Language
TLS	Transport Layer Security
XSPA	OASIS Cross-Enterprise Security and Privacy Authorization

8 REFERENCES

[1] National Institute of Standards and Technology, Developing Trust Frameworks to Support Identity Federations, Internal Report, NISTIR 8149
https://doi.org/10.6028/NIST.IR.8149

[2] Government of Canada, Pan-Canadian Trust Framework (PCTF) Version 1.0
https://github.com/canada-ca/PCTF-CCP

[3] HL7 Version 3 Standard: Privacy and Security Architecture Framework –Trust Framework for Federated Authorization Release 1, May 2018

[4] OAuth 2.0 Token Introspection, October 2015
https://tools.ietf.org/html/rfc7662

[5] OAuth 2.0 Dynamic Client Registration Protocol, July 2015.
https://tools.ietf.org/html/rfc759

[6] National Institute of Standards and Technology, Electronic Authentication Guideline, NIST Special Publication SP 800-63-2, August 2013.
https://dx.doi.org/10.6028/NIST.SP.800-63-2

[7] Vectors of Trust, October 2018.
https://tools.ietf.org/html/rfc8485

[8] Scaling Dynamic Client Registration and Endpoint Discovery for Open APIs – DRAFT, February 2019.
https://docs.google.com/document/d/1yMNGDtB2X_6Maj8MAeVVv3ZI4z0X6LZAo5SisPuT_6k/

[9] OAuth 2.0 Authorization Server Metadata, June 2018
https://tools.ietf.org/html/rfc8414

[10] OAuth 2.0 Dynamic Client Registration Management Protocol, July 2015.
https://tools.ietf.org/html/rfc7592

[11] Pre-OAuth Entity Trust (POET) – DRAFT, October 2017.
https://github.com/TransparentHealth/poet

[12] Security Assertion Markup Language Cross-Enterprise Security and Privacy Authorization (SAML XSPA) v2.0, Working Draft 18, February 2019
https://www.oasis-open.org/committees/download.php/64765/saml-xspa-v2.0-wd18-2019-02-25.docx